



Software Requirement Specification

for Dynamic Log Viewer and Smart Log Analysis

Murat Ekmekçi
Hamza Yılmaz
Ahmed Furkan Akdeniz
Mümin Korcan Aydın

LOGAR by **MİKSER**

1 Introduction

This document is the software requirement specification of the Dynamic log viewing and smart log analysis project mentored by Labris Teknoloji. The SRS is following the criteria and the structure defined on the document “IEEE Recommended Practice for Software Requirements Specifications” published in 1998.

1.1 Purpose

This SRS gives information about the project called “LOGAR”, the requirements of it, the technology and software its based on, capabilities, data and behavioral models, design and data constraints and the timeline of the development phases.

The document is intended for the developers and specialists whom are aware of the logging concept and want to use Logar to have more control on logs.

1.2 Document Conventions

The document is the brief but clear explanation of the dynamic log viewing project, Logar. However, the project is in the very early stages of development. So, even the main motivation or concepts will not change, the versions of the required technologies, the raw interface of the program, data models or the interaction between user and program may change due to the needs of the company or additions and removals by Mikser. The document will be updated according to the statements above.

1.3 Problem Definition

In early stages of every programmer's learning phase, the easiest and the most common way to check whether the program works properly is to use simple printing methods. However, when it comes to understanding the activities of complex systems, particularly in the case of applications with little user interaction, logs are essential.

The best way to investigate logs to understand what happens at the background in a program or system while working, is to use log

viewers. Log viewers come with basic user- interfaces since their only job is viewing and parsing logs but those basic interfaces yield to a mess when it comes to display logs and investigate them. They provide so much detail that not every user can analyze and make a decision about. Besides that even if the registered users always use the same settings, they have to provide those settings each and every time. The communication via users who want to share filtered or raw logs is also missing.

1.4 Project Scope

Mikser intends to implement a web-based dynamic log viewer called "Logar" that will let users to view their logs from multiple sources(remote or local) in real-time with the filtering parameters defined by the user.

This project will be a combination of existing log viewer softwares' most used and best parts without keeping the unnecessary details. Making it as much as simple but functional, users will be able to have their own options defined and stored. While the most common logging types will be in use, for newly developed or undefined log types (by the requests of company), new parsers will be written. Giving users the portability of web-based applications and capability to analyze local logs and remote logs, our program will also communicate user via e-mail in conditions defined by user (such as notifying on fatal log errors or automatic notification at intrusion detection).

1.5 Literature Survey

Of course, there exists various log viewers at the market which are mostly freeware. They have common and basic properties since the scope of application is not very wide. All of them have support for two most known log file types; log4j files and java.util.logging files. They are built just to monitor but not to interfere. They have the ability to show log files by their message level (trace, info, warn, error,...), type, id ,timestamp or etc...

Some of these log viewers are desktop applications such as Otros Log Viewer, Apache Chainsaw, some are web-based applications such as Logio or Android based CatLog.

Otroslogviewer is a software for analyzing an application's logs and stack traces, but does not monitor logs. It requires Java 6 and works from desktop. It can load logs from remote servers or local disks, provides log filters, enables log event searching and listening on a socket. It does not have a installer but a .bat file directly running from user's desktop

Apache Chainsaw is also a desktop based log viewer with a less complicated interface and options. Chainsaw can read log files formatted in Log4j's XMLLayout, receive events from remote locations, read events from a DB. It can even work with the JDK 1.4 logging events.

Logio is a web based log viewer working on browsers like Firefox or

Chrome. It is designed for live monitoring of log files. It is not intended to operate on any other input sources beside files. It is not intended to store, archive or index log messages, nor is it intended for historical or aggregate analysis. But since it has no persistence layer, it has low latency. Server and harvesters will only transmit over the wire if a web client is asking for log data. This improves UI performance while reducing bandwidth consumption.

CatLog is an android based log viewer where it can record, save and mail logs. Its attributes are basic but very useful.

1.6 Definitions and Abbreviations

Definitions:

Guest User : The person who just enters the web site unregistered and can check local logs and remote sources.

Registered User : The person who is registered to the site and able to use whole functions of it.

Company: Labris Teknoloji, the mentor of Mikser's log viewer project.

Logar: Name of the project.

Mikser: Name of the project group

Log4j: A logging library for java, created by Ceki Gülcü

XmlLayout: A layout for output files created by logger program.

Abbreviations:

SRS: Software Requirements Specifications

DB: Database

GUI: Graphical User Interface

JVM: Java Virtual Machine

IDS: Intrusion Detection System

1.7 References

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications

- <http://code.google.com/p/otroslogviewer/>

- <http://logging.apache.org/chainsaw/index.html>

- <http://logio.org/>

- <http://nolanlawson.com/apps/#catlog>

- About log formats

<http://logging.apache.org/log4j/2.x/>

<http://docs.oracle.com/javase/1.4.2/docs/api/java/util/logging/XMLFormatter.ht>

[ml](#)

- <http://www.labristeknoloji.com/>
- The Complete Log4j Manual by Ceki Gülcü, 2002
- Project's development and distribution website
<https://senior.ceng.metu.edu.tr/trac/mikser>

2 Overall Description

2.1 Product Perspective

Logar is a project that will be developed for internet, allowing programmers, developers and specialists to investigate logs. It will be free to register into website and the source code will be shared. There are numerous programs on internet that can be used, few of them have advanced features. Mostly they have the basic log viewing property in common.

Mikser will take all the advantages of these log viewers and combine them in our fast working, web based program - Logar. However there are some complex features and attributes of these programs too which we will avoid since the general use of those is less likely and they will create an interface mess more likely.

Mikser hopes to get ideas from developers in recent future, encouraging them thinking out of the box and help us think the same way too. We will shape our product's final design considering the future recommendations and suggestions.

2.2 Product Features

Completed version of Logar will provide:

- let users signup to the website.
- read logs from local disks or remote sources.
- define filter settings and save them for later use.
- define a log file's path in local disk and save the path for later use.
- define a remote source connection and save it for later use.

- parse logs according to the filters.
- e-mail a raw log or filtered version of a log.
- compare two logs.
- view logs on the website dynamically.
- listen to the sockets
- automatic mailing on fatal level detection
- display logs in a highlighted manner
- language support for Turkish and English

2.3 User Classes and Characteristics

Logar will be designed for two major types of user classes:

1. General users who will visit our site to cover their needs

- Guest User, as we like to call them, "One Time Users", are software developers who use logging utilities rarely, and needed Logar to check a log in their harddrive just for once or twice.

- Registered User are more experienced loggers and use logging utilities more often. They would like to investigate logs deeper.

2. Open source developers and contributors

- People who are also able to write down their own logviewers but searching for ideas.

- Developers who are interested in helping us since its a free source code.

- Our mentor, Labris Teknoloji's developers where they would like to take a look and contribute.

2.4 Operating Environment

Logar is platform independent but runs on java based technologies (recommended complete software update). An internet connection is a must since Logar works online.

2.5 Design Constraints, Assumptions, Dependencies

To use the website's full attributes, developers have to sign up. Users must have permission to access remote sources which they would like to inspect logs from. Also for local logs on harddrives, the operating system's permission must be granted or user should change the permission manually.

A registered user can compare up to two entries at the same time but no more. Also the log file's layout must be compatible with the project's parser definition.

We strongly recommend an update for java software and Javascript must be enabled for web browser.(JVM 1.6 or higher)

2.6 User Documentation

The project's documentation will be available at

- <https://senior.ceng.metu.edu.tr/trac/mikser>

for Mikser developers and Course instructors. A basic user manual will be put into the website about January 2013.

3 Usage Cases

3.1 User Profiles

3.1.1 Guest User

- can sign-up to the website
- access local disk
- investigate a log in local disk or remote source
- use filters to narrow down entries
- clear filters by one button
- clear log panel by one button
- compare entries

3.1.2 Register User

- login to the system by entering e-mail address and password
- access local disk
- investigate a log in local disk
- access remote source if permission granted
- use filters to narrow down entries

- clear filters by one button
- clear log panel by one button
- compare entries
- save filter settings
- save a log's path in local disk
- save remote source's host, user, port, password
- load filter settings
- load a log's path in local disk
- load remote source's host, user, port, password and connect
- dynamic watching of log
- pause and continue watching
- e-mail the entire log file as an attachment
- e-mail the filtered messages as an attachment
- listen to the sockets
- get automatic notification from system when fatal errors occur on listening procedure
- quit account and sign-in as different user

3.2 Usage Scenarios

Here are some basic usage scenarios.

3.2.1 Newbie programmer

- they can use the program as some kind of a error tracker
- they can investigate logs from their programming environment
- programmer may sign-up for further investigations and e-mail the log to their co-programmer friends in a project.

3.2.2 Software developer in a Bank

- they can use the program as a security measure
- they can investigate logs in a simple manner provided by Logar
- they can listen to the sockets
- they can detect movements in logs at certain accounts.
- Automatic e-mails will be sent to them when a fatal error occurred where the definition of fatal sometimes mean

intruders.

4 Data Model

4.1 Data Dictionary

- **User:** the registered person who uses Logar.

Mail: user's e-mail. Used as primary key.

Password : user defined string for login to the system.

- **Filters:** attributes letting user narrow down log entries with predefined settings

Mail: user's e-mail. Used as primary key.

StartID: a number used to filter ids smaller than itself

EndID: a number used to filter ids bigger than itself

MatchingString: string to match log entries

StartDate: a date used to filter dates before itself

EndDate: a date used to filter dates after itself

Level: a string to get rid of the levels below itself

Regex: a regular expression to match

Class: caller class' name

Logger: caller logger's name

- **RemoteSource:** Connection parameters to access remote servers for later uses

Mail: user's e-mail. Used as primary key.

Hostname: name of the host to connect

Username: username to connect

PortNumber: port number for listening

RemotePass: password if needed

- **LocalPath:** Full path name needed for later uses in local accesses to logs

Mail: user's e-mail. Used as primary key.

FilePath: path name

FileName: log file's name

- **SavedMessage:** User can e-mail entire log or a filtered part of it

saved here

<i>Mail:</i>	user's e-mail. Used as primary key.
<i>ID:</i>	ID number of the message
<i>Time:</i>	the time its created
<i>Message:</i>	message created by the logger system
<i>Logger:</i>	name of the logger
<i>Level:</i>	degree of the log
<i>ToWhom:</i>	e-mail address to send

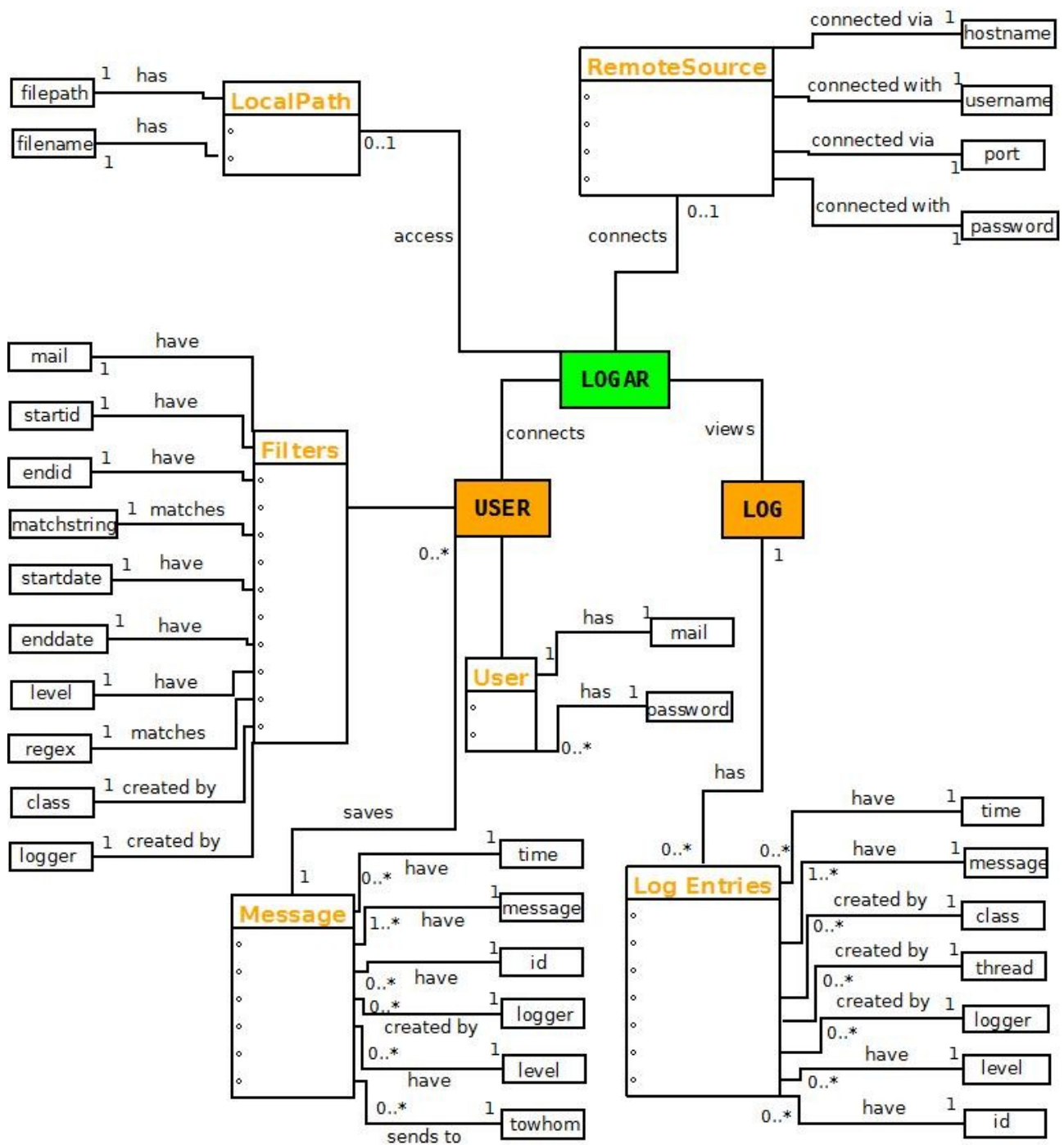
-UserRecentXMLFilterTable: Website's way to store data fast.

<i>RecentLog:</i>	name of the log
<i>RecentTimeStart:</i>	filter to eliminate time below it
<i>RecentTimeEnd:</i>	filter to eliminate time above it
<i>RecentLevel:</i>	filter the levels below it
<i>RecentIDStart:</i>	filter to ids below it
<i>RecentIDEnd:</i>	filter to ids above it
<i>RecentString:</i>	matching string
<i>RecentRegex:</i>	matching regular expression

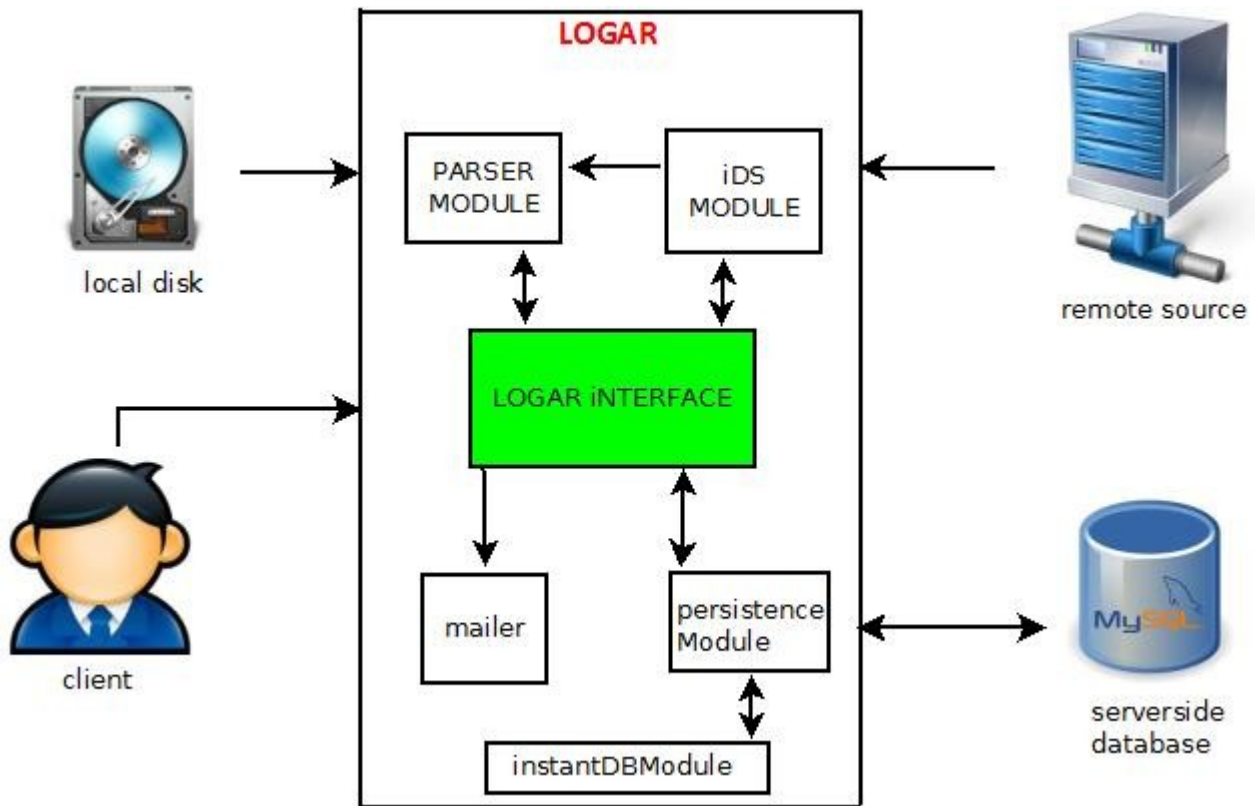
-LogEntryXMLFile: All the entries goes here

<i>EntryID:</i>	id number
<i>EntryTime:</i>	creation time
<i>EntryMessage:</i>	message in it
<i>EntryClass:</i>	caller class
<i>EntryMethod:</i>	caller method
<i>EntryThread:</i>	caller thread
<i>EntryLogger:</i>	<i>name of the logger</i>
<i>EntryLevel:</i>	<i>degree of the level</i>

4.2 Data ER Diagram

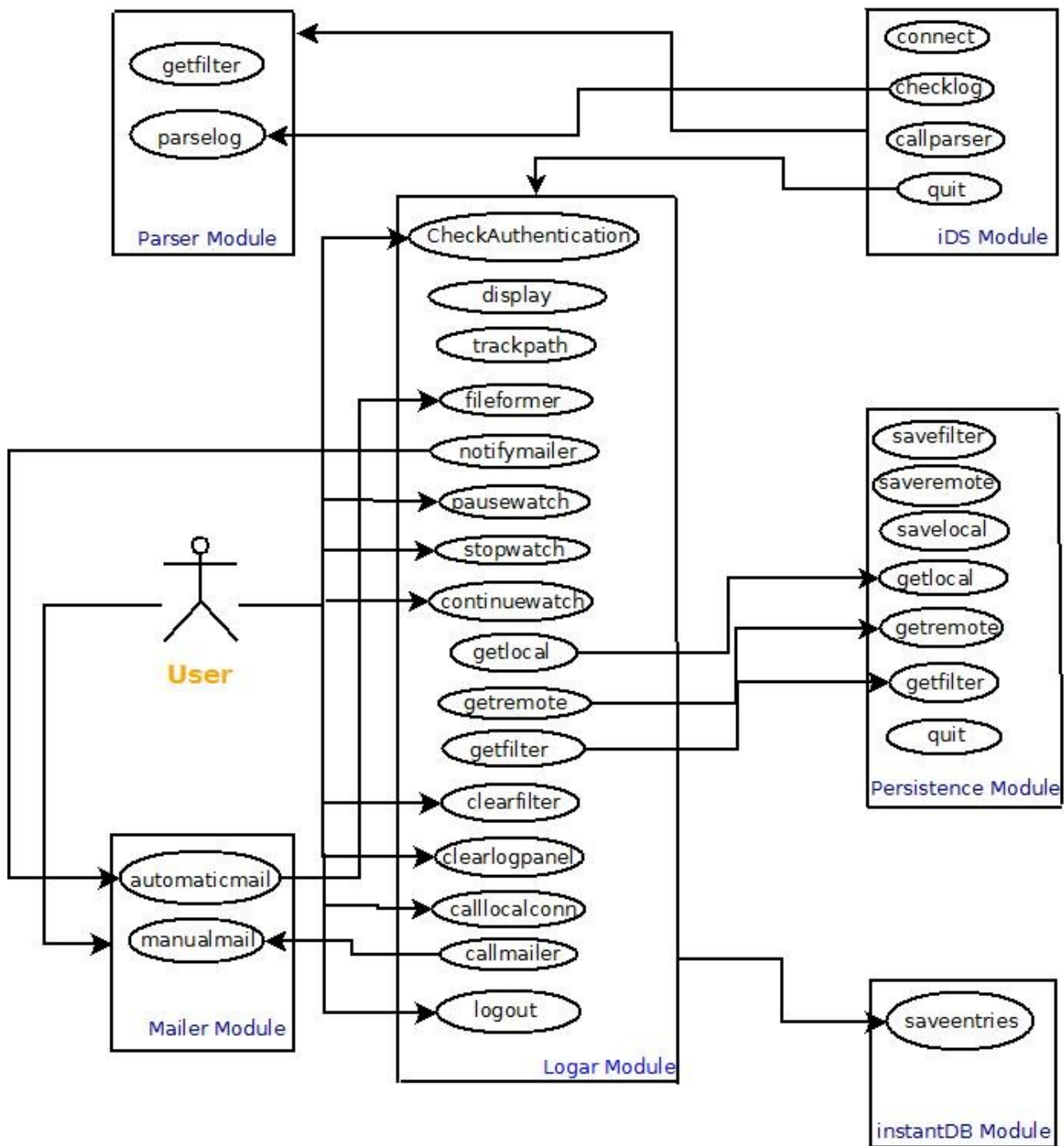


5.1 Functional Models



general layout of the Logar

This is the general layout of the project Logar. It consists of six modules. Everything starts with the request of client. Client(user) interacts system with Logar Interface. The Logar Interface can reach any module in the system as expected and is one of the complex modules. Parser Module, another complex module, will be designed to filter logs. IDS module, last of the complex modules, is the intrusion detection system that is responsible for remote connections. Persistence Module is taking care of save and load jobs for user defined settings while InstantDB Module is responsible for recording vast amount of data on xml files. Finally, last of the modules, Mailer module is doing what its name requires, responsible for manual and automatic mailing.



Use Case Diagram of Logar

5.2 Function Descriptions

5.2.1 Logar Module

5.2.1.1 CheckAuthentication

Parameters given by user, this function connects to system's database, checks if the user is registered before.

5.2.1.2 Display

Logar module has static and dynamic parts in main page. Static parts such as filtering options changes only when user interferes with them however dynamic parts are changing as new logs and entries come. Display is responsible for these matters. User cannot the inner work style or the ways of showing logs.

5.2.1.3 TrackPath

The function is responsible for the local connections and tracking, attaching files to the system.

5.2.1.4 FileFormer

Takes saved messages, put them in a file form(*.txt or *.log)

5.2.1.5 NotifyMailer

If logar module takes a notification from ids module about intrusion or fatal error,this function calls AutomaticMailer.

5.2.1.6 PauseWatch

Bind to the pauseWatch button in main page, it will stop displaying of new entries but at the background log checking will continue.

5.2.1.7 Stopwatch

Bind to the stopWatch button in main page, both displaying and background log watching will stop.

5.2.1.8 ContinueWatch

Incase of pauseWatch button pressed, ContinueWatch function bind to the continueWatch button in main page, will let main page order work as before.

5.2.1.9 GetLocal

Connects to the database, gets user's predefined local path and filename.

5.2.1.9 GetRemote

Connects to the database, gets user's predefined hostname, username, port number and password for remote connection.

5.2.1.10 GetFilter

Connects to the database, gets user's predefined filter settings.

5.2.1.11 ClearFilter

With one button click, all filter settings on main page comes to default options.

5.2.1.12 ClearLogPanel

With one button click, log panel- where the entries displayed dynamically line by line- will be cleared.

5.2.1.13 CallLocalConn

System's function responsible for getting log file to the system itself when GetLocal function is used.

5.2.1.14 CallMailer

Let user e-mail manually.

5.2.1.15 Logout

User logout of the system, all the displaying and connections stopped.

5.2.2 Persistence Module

5.2.2.1 SaveFilter

Parameters given by user, this function connects to system's database, save the filter options. Options will be also saved into a XML file for instant use, to prevent repetitive database connections.

5.2.2.2 SaveRemote

Parameters given by user, this function connects to system's database, save the data needed to connect remote source .

5.2.2.3 SaveLocal

Parameters given by user, this function connects to system's database, saves local file path and name.

5.2.2.4 GetLocal

Connects to the database, gets user's predefined local path and filename.

5.2.2.5 GetRemote

Connects to the database, gets user's predefined hostname, username, port number and password for remote

connection.

5.2.2.6 GetFilter

Connects to the database, gets user's predefined filter settings.

5.2.3 IDS Module

5.2.3.1 Connect

Connects to the remote source with user given form data.

5.2.3.2 CheckLog

Rapid check of logs.

5.2.3.3 CallParser

Works together with Parser Module. Use filter settings to filter logs.

5.2.3.4 Quit

Stops listening to the socket, closes all connections.

5.2.4 Parser Module

5.2.4.1 GetFilter

gets filter settings from the userrecentxmlfiltertable.

5.2.4.2 ParseLog

Most important function of all. Reads logs interpret them with the given filter settings, pass result to the display.

5.2.5 InstantDB Module

5.2.5.1 SaveEntries

Saves log entries into the logentryxmlfile to be able to process them quickly.

5.2.6 Mailer Module

5.2.6.1 AutomaticMail

Called by NotifyMailer, it sends an e-mail automatically to inform user about a fatal situation or intrusion occurred.

5.2.6.2 ManualMail

an entire log User may use this function to send an e-mail to himself
file or a pack of filtered entries in a txt file.

6 Non-functional Requirements

6.1 Performance Requirements

Logar is web based project. It may require several users to connect simultaneously. This will not be such a problem since we are not Facebook! But assuming in the recent future that everybody is using log viewers and especially Logar, we will take our precautions. But for users not to get any frustrations, a faster internet connection, complete update on browsers, enabled javascript on browser is necessary.

Here the main problem may occur in different ways. The web server's capabilities, bandwidth, efficiency of the code, the amount of messages per second, the way we design database etc. Mikser may change design to improve performance as needed.

6.2 Design Constraints

6.2.1 Safety

Logar will change neither local file nor remote files. In case of trying to access files that Logar unfamiliar with, user will get warning in log panel. System files is also in this class so no change will be done on them too. Further testing will be done for system crashes.

6.2.2 Security

To prevent any spy programs get socket's hostname, username, port number and especially password, we will use them as encrypted. It will be done in the same way when storing them into database. Users will have personal account so no one else will be able to see others' personal logs.

6.2.3 Software Quality

Logar works as one single page. It will not contain messy links, buttons that is not known what for or complicated technical things. While expecting experienced users in our website, we also encourage beginners to sign-up so they can fully benefit from their programs by examining the logs

6.2.5 Accessibility

Since its a web based program, developers can use the system 24/7, anywhere with a computer and internet access.

7 Behavioral Model

7.1 Description for Software Behavior

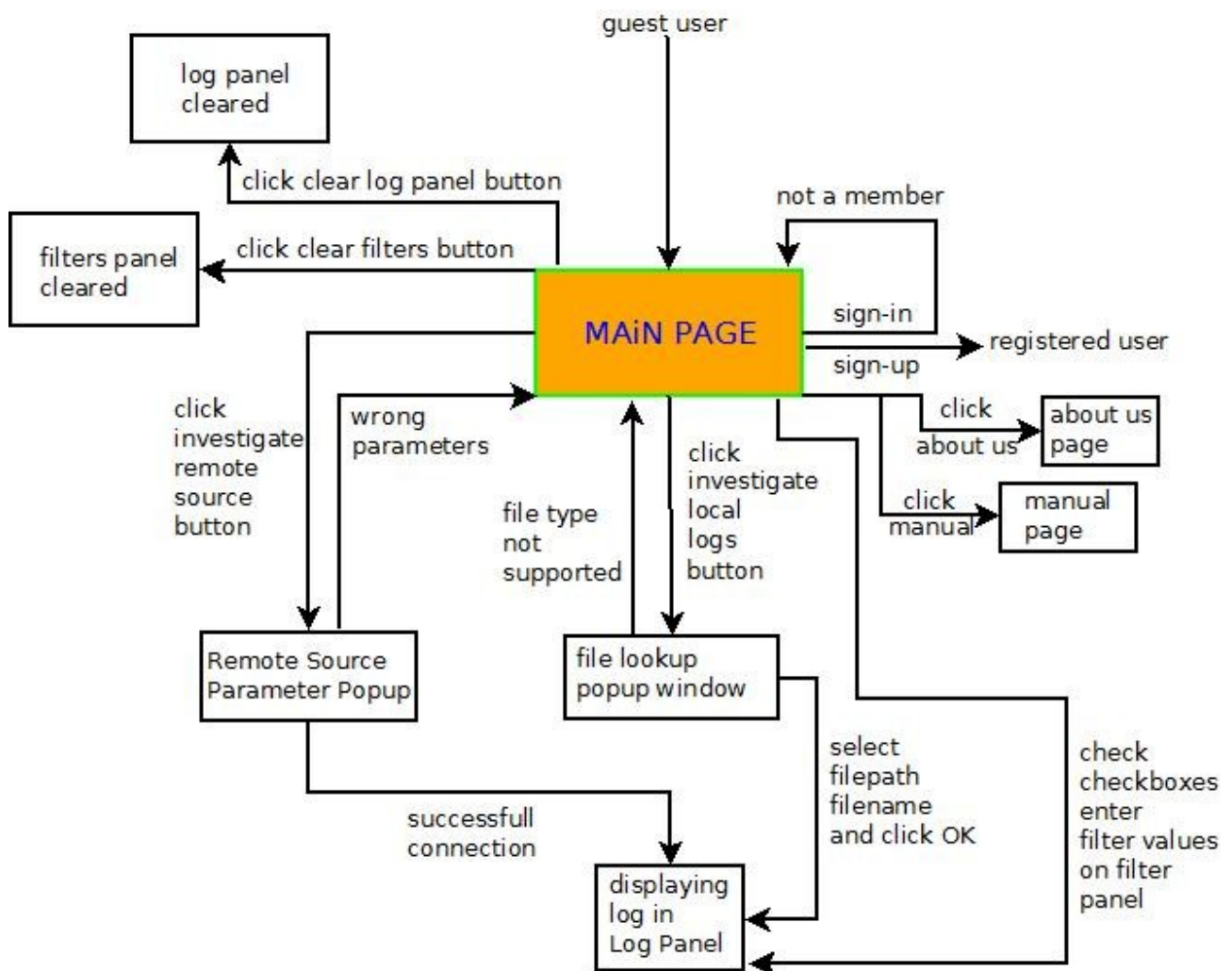
Since there are two types of users- guest and registered-, then two types of behavior exist. The plan to divide user types into two, has come with the idea of distinguishing every day user from "one time" users. We wanted to implement parts that are not present at other log viewers such as personal preferences so this requires users to login to let us keep their data in database.

Guest Users can view Logar interface as they enter the site. They can read information about us and check Logar's manual. They can sign-up if they want to(we encourage them). Also when it comes to the real functions of the site, they can examine logs in their harddrives by giving full pathname or remote sources by providing connection parameters. This kind of users are also allowed to use filters, clear filter settings or log panel.

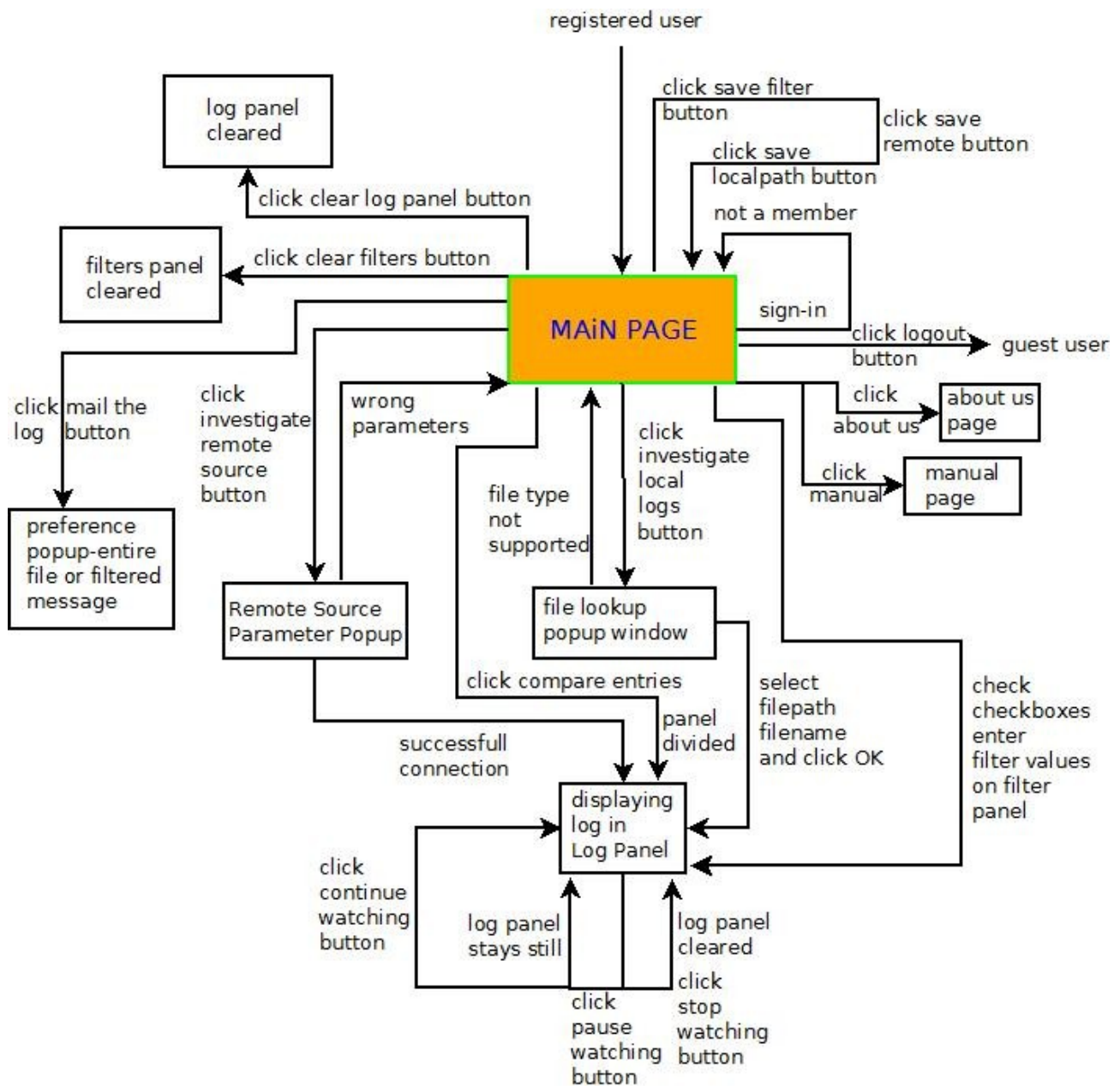
Registered Users are able to do everything guest users can. Besides those, they can save their filter settings fully local path's name, remote sources parameters. This lets them to load these saved attributes for ease of usage. They may stop, pause or continue watching log entries listed in log panel. Differing from guest users, registered users can e-mail filtered messages or entire log file and they can get automated e-mails from system.

Also we let them compare 2 logs side by side too. Finally they can quit their session happily ever after.

7.2 State Transition Diagrams



state transition diagram of guest user



state transition diagram of registered user

8 Planning

8.1 Team Organization

We believe that controlled centralized (CC) team organization is the best model to fit our needs, team structure, and project's properties. The reasons for our choosing Controlled Centralized team organization are listed below:

- In order to make maximum utilization of labor of division, at times we should work with in subgroups. With the Controlled Centralized approach sub grouping can be used effectively.
- To meet software design requirement we will be using modularity at its maximum extent. With modularity we gain simplicity in all phases of project development. And also the maintenance will be easier.
- Controlled Centralized team organization has been found to produce fewer defects than the other team structures. Since we are a new company, we want to gain prestige in the market, developing highly reliable software solutions is the main concern in our software development.

By places in organization;

Project Manager : *Hamza Yılmaz*

Head Developer : *Murat Ekmekçi*

Developers : *Ahmet Furkan Akdeniz, Mümin Korcan*

Aydın

By modules working on;

Logar Interface : *Hamza Yılmaz, Murat Ekmekçi, Mümin Korcan*

Aydın

Persistence Module: *Ahmet Furkan Akdeniz, Mümin Korcan Aydın*

IDS Module : *Hamza Yılmaz, Ahmet Furkan Akdeniz*












Parser Module : *Murat Ekmekçi, Mümin Korcan Aydın*

InstantDB Module: *Murat Ekmekçi, Hamza Yılmaz*

Mailer Module: *Ahmet Furkan Akdeniz*

Even though we plan to subgroup for different modules, we will help each other and communicate often to keep program intact. This way members will be able to change places in organization. Also when one of the modules is completed, developers may contribute to other modules and solve design and implementation problems together.

8.2 Project Schedule

		Name	Duration	Start	Finish
1		▢PLANNING	8 days?	11.10.2012 08:00	22.10.2012 17:00
2		project survey	3 days	11.10.2012 08:00	15.10.2012 17:00
3		decision of roles	1 day?	11.10.2012 08:00	11.10.2012 17:00
4		quality and configuration plan	1 day?	15.10.2012 08:00	15.10.2012 17:00
5		project proposal	3 days	18.10.2012 08:00	22.10.2012 17:00
6		milestone	0 days	22.10.2012 17:00	22.10.2012 17:00
7		▢ANALYSIS	8,875 d...	01.11.2012 09:00	13.11.2012 17:00
8		requirement analysis	3 days	01.11.2012 09:00	06.11.2012 09:00
9		determine problem constraints	1 day?	01.11.2012 09:00	02.11.2012 09:00
10		data modelling	2 days	06.11.2012 09:00	08.11.2012 09:00
11		functional modelling	2 days	06.11.2012 09:00	08.11.2012 09:00
12		behavior modelling	2 days	06.11.2012 09:00	08.11.2012 09:00
13		data dictionary	1 day?	06.11.2012 09:00	07.11.2012 09:00
14		SRS report	3 days	08.11.2012 09:00	13.11.2012 09:00
15		milestone	0 days	13.11.2012 17:00	13.11.2012 17:00
16		▢DESIGN	12,875 ...	15.11.2012 09:00	03.12.2012 17:00
17		architectural design	10 days	15.11.2012 09:00	29.11.2012 09:00
18		logar interface design	10 days	15.11.2012 09:00	29.11.2012 09:00
19		component level design	10 days	15.11.2012 09:00	29.11.2012 09:00
20		initial design report	10 days	15.11.2012 09:00	29.11.2012 09:00
21		interface design first version	8,875 d...	20.11.2012 09:00	30.11.2012 17:00
22		IDS design first version	7,875 d...	20.11.2012 09:00	29.11.2012 17:00
23		persistence layer design	3,875 d...	26.11.2012 09:00	29.11.2012 17:00
24		detailed design report	2,875 d...	29.11.2012 09:00	03.12.2012 17:00
25		milestone	0 days	03.12.2012 17:00	03.12.2012 17:00
26		▢PROTOTYPE DEVELOPMENT	32,875 ...	05.12.2012 09:00	18.01.2013 17:00
27		design improvements	10 days	05.12.2012 09:00	19.12.2012 09:00
28		requirement and design report update	5 days	24.12.2012 08:00	28.12.2012 17:00
29		coding prototype	20 days	17.12.2012 09:00	14.01.2013 09:00
30		prototype demo	4,875 d...	14.01.2013 09:00	18.01.2013 17:00

